SPARQL Query Recommendations by Example

Conference or Workshop Item

SPARQL Query Recommendations by Example

Carlo Allocca, Alessandro Adamou, Mathieu d'Aquin, and Enrico Motta

Knowledge Media Institute, The Open University, UK,
{carlo.allocca,alessandro.adamou,mathieu.daquin,enrico.motta}@open.ac.uk

Abstract. In this demo paper, a SPARQL Query Recommendation Tool (called SQUIRE) based on *query reformulation* is presented. Based on three steps, *Generalization*, *Specialization* and *Evaluation*, SQUIRE implements the logic of reformulating a SPARQL query that is satisfiable w.r.t a *source* RDF dataset, into others that are satisfiable w.r.t a *target* RDF dataset. In contrast with existing approaches, SQUIRE aims at recommending queries whose reformulations: `i)` reflect as much as possible the same intended meaning, structure, type of results and result size as the original query and `ii)` do not require to have a mapping between the two datasets. Based on a set of criteria to measure the similarity between the initial query and the recommended ones, SQUIRE demonstrates the feasibility of the underlying *query reformulation* process, ranks appropriately the recommended queries, and offers a valuable support for query recommendations over an unknown and unmapped target RDF dataset, not only assisting the user in learning the data model and content of an RDF dataset, but also supporting its use without requiring the user to have intrinsic knowledge of the data.

1 Introduction

One of the main aspects that characterises Linked Open Data (LOD) is *Heterogeneity*: it is not hard to find RDF datasets that describe overlapping domains using different vocabularies [11]. A long-standing challenge raised by this state of affairs is related to the *access* and *retrieval* of data. In particular, a common scenario that is playing a central role to accomplish a number of tasks, including integration, enriching and comparing data from several RDF datasets, can be described as follows: given a query Q_o (e.g. `Select distinct ?mod ?title where { ?mod a ou:Module. ?mod dc:title ?title }`[1]) formulated w.r.t a *source* RDF dataset D_s (e.g. D_{ou}= `http://data.open.ac.uk/query`), we need to reformulate it w.r.t another similar *target* RDF dataset D_t (e.g. D_{su}=`http://sparql.data.southampton.ac.uk`). Achieving this goal usually involves quite intensive and time consuming ad-hoc pre-processing [12]. In particular, it requires spending time in exploring and understanding the target RDF dataset's data model and content, and then, iteratively reformulating and testing SPARQL queries until the user reaches a query formulation that is right for his/her needs [3]. Reformulating a query over many RDF datasets can be very laborious but, if aided by tool support that recognises similarities and provides prototypical queries that can be tested without the user's prior knowledge of the

[1] Select the names of the modules available at The Open University.

dataset, the time and effort could be significantly reduced. In this demo paper, we propose a novel approach and a tool (called SQUIRE) that, given a SPARQL query Q_o that is satisfiable w.r.t a *source* RDF dataset (D_s), provides query recommendations by automatically reformulating Q_o into others Q_{r_i} that are satisfiable w.r.t a *target* RDF dataset (D_t). In contrast with existing approaches (see Section 2), SQUIRE aims at recommending queries whose reformulations: i) reflect as much as possible the same intended meaning, structure, type of results and result size as the original query and ii) do not require to have an ontology mapping and/or instance matching between the datasets. Based on a set of criteria to measure the similarity between the user-provided query Q_o and the recommended ones Q_{r_i}, we have prototyped our approach. Demo session attendants will have the opportunity to experiment with SQUIRE over real-world SPARQL endpoints, thus demonstrating the feasibility of the underlying *query reformulation* and *query recommendation* processes. The paper structure is as follows: Sec. 2 discusses existing works, Sec. 3 details the SQUIRE's approach and its implementation. Finally, Sec. 4 concludes and points out future research.

2 Related Work

To the best of our knowledge, there is no other study investigating SPARQL query recommendations over unmapped RDF datasets that take user queries into account. On the contrary, several solutions exist to address the issue of SPARQL query rewriting for implementing data integration over linked data. For instance, [7] devised a query rewriting approach that makes full use of schema mapping, whereas [2] relies on an explicit ontology alignment between the source D_s and the target D_t. Similarly, [9] described a method for query approximation where the entities appearing in the query can be generalized w.r.t an given ontology mapping. Moreover, several systems have been proposed, with very good achievements, to support users with no knowledge on SPARQL or RDF to build appropriate queries from scratch. Just to mention a few, Sparklis [4], QUICK [12], QueryMed [10] are designed on a query building process that is based on a guided interactive questions and answers. The authors of RDF-GL [6] designed a method based on a visual query language where a query can be viewed in a natural language-like form and in a graphical form. On the same line, but hiding the SPARQL language syntax, SparqlFilterFlow [5] and SPARQLViz [1] proposed an approach based on visual interface where the queries can be created entirely with graphical elements. Closer to our goal, [3] aims at alleviating the effort of understanding the potential use of an RDF dataset by automatically extracting relevant natural language questions that could be formulated and executed over it. Although all the above studies were useful for us as they contribute interesting elements to build on, they are mainly driven by a context where the user is not familiar with the underlying technologies (which is not our case) and having in mind the goal that semantic access and retrieval of data can be made more usable through an appropriate natural language based systems. In contrast, we are focusing on a method to make SPARQL query recommendations by reformulating a user query for accessing and retrieving data from unmapped RDF datasets.

3 Method and Implementation

To achieve our goal, SQUIRE proposes and implements a mechanism based on three steps: *Generalization*, *Specialization* and *Evaluation*. To present each of them, let us consider the case in which we want to build recommendations for the example query Q_{ou} but w.r.t. the Southampton University RDF dataset D_{su}.

Generalization aims at generalizing the entities (classes, properties, individuals and literals) of Q_o that are not present in D_t into variables (marked as *template variables*)[2]. By applying this step, we build what we called the *Generalized Query Template* (GQT). Back to the query Q_{ou}, the GQT is obtained from it by turning the entities `ou:Module` as a class and `dc:title` as a datatype property into two template variables, that is `?ct1` and `?dtp1`, respectively. The result is shown in the root node of the tree in Fig. 1.

Specialization aims at specializing consistently the obtained GQT by applying two main operations: (a) *Instantiation* (I) instantiates consistently a template variable with a corresponding concrete value that belongs to D_t (e.g. we instantiate `?ct1`[`?dtp1`] over each class [datatype property] of D_{su}); and (b) *Removal* (R) deletes an entire triple pattern from the Q_o's GQT. We called the output of this step *Specialized Query Tree*. Fig. 1 shows a part of it for Q_{ou}.

Evaluation. As a result of the previous two steps, each tree node is considered to be a reformulated query that is a candidate for recommendation. However, some of them are more "similar" to the original one than the others. Thus, the main question was: how can we capture and compute such similarity to provide a score-based ranking? Being in accord with [8] that there is no universal way of measuring the distance and/or similarities between two formal queries, we based our approach on a linear combination of the following criteria[3]: (a) *Result Type Similarity* aiming at measuring the overlap of the types of results (URI or literal) between Q_o and any recommendations Q_{r_i}; (b) *Query Result Size Similarity* aiming at measuring the result size rate (normalized w.r.t datasets sizes) between Q_o and any recommended one Q_{r_i}; (c) *Query Root Distance* aiming at measuring the cost of each applied operation from the root node to the one containing the recommended Q_{r_i}. It takes into account the distance-based matching of the replaced entities and the structure (as a set of triple patterns) between Q_o and any recommended Q_{r_i}; and (d) *Query Specificity Distance* aiming at measuring the distance between Q_o and any recommended Q_{r_i} based on the sets of variables (total shared variables / total variables).

A screenshot of the implemented tool is shown in Fig 2. Basically, SQUIRE allows the user to (1) refer to a source RDF dataset, either as an RDF file or as the URL of a SPARQL endpoint; (2) write down the query Q_o w.r.t D_s and (3)

[2] It is a query variable that has been consistently indexed with natural numbers and named according to its type. We used ct, it, opt, dpt and lt, for class, instance, object property, data type property and literal, respectively.

[3] The weights are options given to the user to set based on their preferences.

specify the target RDF dataset. Once the user clicks on the *Recommend* button, SQUIRE executes the method described above and returns a list of scored recommended queries, sorted high-to-low. Another distinctive characteristic of SQUIRE is that the recommended queries not only are expressed in terms of the target dataset, but also are guaranteed to be satisfiable (i.e. the result set is not empty) and can therefore be used to access and retrieve data from the target dataset.

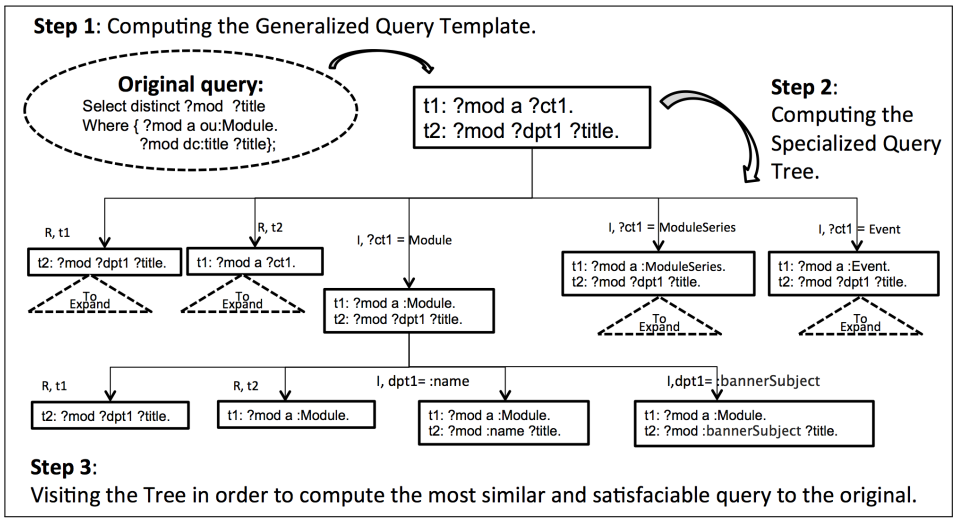

Fig. 1. Part of the *Specialized Query Tree* by applying the two operations.

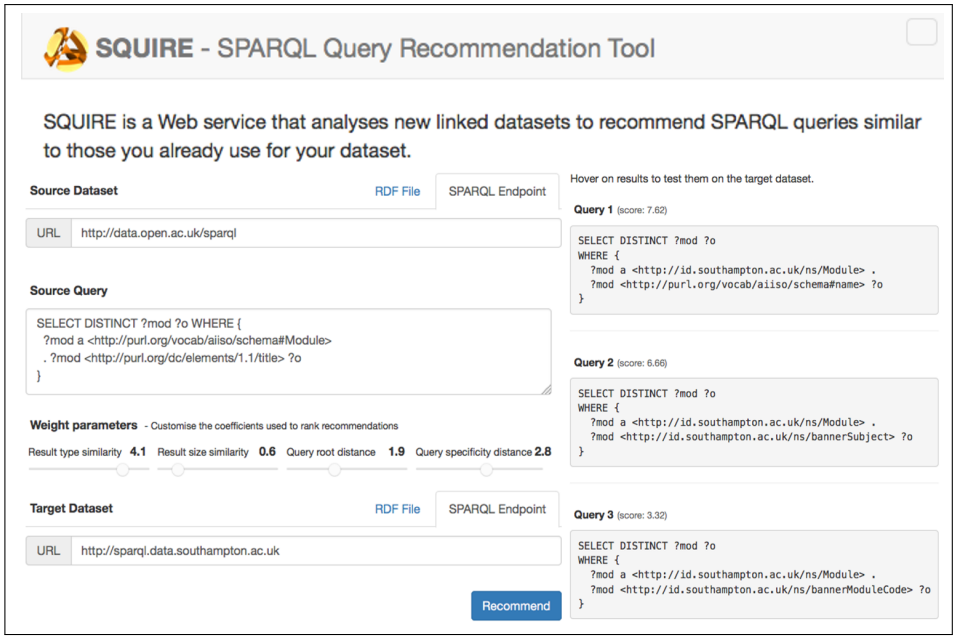

Fig. 2. SQRT prototype screenshot.

4 Conclusion and Discussion

SQUIRE, as an approach and a tool, enables SPARQL query recommendations by reformulating a user query that is satisfiable w.r.t a *source* RDF dataset D_s, into others that are satisfiable w.r.t a *target* (and unmapped) RDF dataset D_t. One of the advantages of SQUIRE is that not only it helps learning the data model and content of a dataset, which usually requires a huge initial effort, but also enhances their use straightforwardly without the user's prior knowledge. Indeed, the problem is not fully solved. One of the aspects we have planned to investigate is the case where the reformulation is based on other types of operations (e.g. adding a triple pattern, or more generally replacing a graph pattern with another one, and so on). Moreover, we want to extend this work in such a way that covers, apart from SELECT (which is the main focus here), other types of queries such as DESCRIBE, CONSTRUCT and ASK. Finally, we believe that the outcomes of research on SPARQL query profiling can be combined with ours to improve the corresponding approaches.

Acknowledgements. This work was supported by the MK:Smart project (OU Reference HGCK B4466).

References

1. J. Borsje and H. Embregts. Graphical Query Composition and Natural Language Processing in an RDF Visualization Interface. *E.S. of E. and B.,Univ.,Rott.*, 2006.
2. G. Correndo, M. Salvadores, I. Millard, H. Glaser, and N. Shadbolt. SPARQL Query Rewriting for Implementing Data Integration over Linked Data. In *Proceedings of the 2010 EDBT/ICDT Workshops*, EDBT '10, NY, USA, 2010. ACM.
3. M. d'Aquin and E. Motta. Extracting Relevant Questions to an RDF Dataset Using Formal Concept Analysis. In *Proc. of the 6th K-CAP*, USA, 2011.
4. S. Ferre. Sparklis: An Expressive Query Builder for SPARQL Endpoints with Guidance in Natural Language. *Sem. Web: Inter., Usab., App.*, 2016. To appear.
5. F. Haag, S. Lohmann, and T. Ertl. SparqlFilterFlow: SPARQL Query Composition for Everyone. In *11th ESWC 2014*, May 2014.
6. F. Hogenboom, V. Milea, F. Frasincar, and U. Kaymak. RDF-GL: a SPARQL-based graphical query language for RDF. In *Em. Web Intel.: Adv. Info. Re.* 2010.
7. K. Makris, N. Bikakis, N. Gioldasis, C. Tsinaraki, and S. Christodoulakis. Towards a Mediator Based on OWL and SPARQL. In *Proc. of the 2Nd*, WSKS '09, 2009.
8. F. Picalausa and S. Vansummeren. What Are Real SPARQL Queries Like? In *Proceedings of*, SWIM '11, pages 7:1–7:6, New York, NY, USA, 2011. ACM.
9. B. R. K. Reddy and P. S. Kumar. Efficient approximate SPARQL querying of Web of Linked Data. In *URSW*, CEUR Workshop Proc. CEUR-WS.org, 2010.
10. O. Seneviratne. QueryMed: An Intuitive SPARQL Query Builder for Biomedical RDF Data, 2010.
11. Y. Tzitzikas, C. Alloca, C. Bekiari, Y. Marketakis, P. Fafalios, M. Doerr, N. Minadakis, T. Patkos, and L. Candela. Integrating Heterogeneous and Distributed Information about Marine Species through a Top Level Ontology. In *MTSR'13*.
12. G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl. From Keywords to Semantic queries-Incremental Query Construction on the Semantic Web. *Web Sem.*, 7(3):166–176, September 2009.